VxWorks 1553 Driver

Testing options

Tue, Jun 23, 1998

Test environment

A test station for exercising the new 1553 driver code running under the VxWorks operating system kernel is located in the lab area on level 3 in the D0 assembly building. The Motorola 162 CPU board runs VxWorks and uses a 1553 controller board to communicate with a D0 Rack Monitor Remote Terminal (RT). The 1553 hardware is identical to that used for the D0 slow controls for the past 10 years. Connection to the VxWorks system for testing is done via telnet to Internet node `d0sil004.fnal.gov`. The development computer system that supports the files needed for VxWorks-based development is `d0chb.fnal.gov`.

Login to the test node via

```
telnet d0sil004.fnal.gov
```

The username is `vxworks`.
The password is known to authorized users.

The test node may be rebooted by typing ctl-x. (This can be too easy to do!)

Preparation of the testing environment

Enter the following commands at the VxWorks telnet prompt:

```
cd “~goodwin”
ld <drv1553.o
ld <drv1553Command.o
moduleShow
sysIntEnable 5
drv1553Init
drv1553Define 0,0x28,10,0x200000,0x2000,0x1000
```

The various source and object files are kept in `d0chb` under the username `goodwin`. The `drv1553.c` file (whose corresponding object file is `drv1553.o`) contains most of the driver source code. The test routine `drv1553Command` is maintained in a separate source file. The two `ld` commands download the object files into the VxWorks system, which accomplishes the required linking automatically. The `moduleShow` command lists the modules now known to the system. Next the level 5 interrupt is enabled from the IRQ5 line on the VMEbus, which is selected on the controller board via a set of three switches. The 1553 driver package is initialized, and one instance of the driver is defined, that for controller #0, whose memory space is located in 24-bit VMEbus standard address space at `0x200000`, a base address also selected by switches on the controller board. Each controller occupies 64K bytes of memory space; however, some may be populated with only 16K bytes of memory chips. Each controller board actually includes two controllers. In this case, controller #1 uses 64K bytes of memory space at address `0x210000`.

The `drv1553Define` command establishes controller #0 as using interrupt vector `0x28`, a limit of 10 entries in the message queue that is used to pass jobs to the driver, the base address of `0x200000`, an offset of `0x2000` bytes to the start of the area of memory used for building the necessary 1553 command block structures necessary to carry out a job. The value `0x1000` specifies that 4K bytes of memory are available for this purpose. In summary, the area from `0x202000-0x202FFF` will be used for building command blocks as needed.

Driver operation

With the installation of each controller, a task is intialized that awaits the request for jobs to be processed via a message queue. A separate message queue is used for each controller. When an entry is read from the message queue, the required number of 1553 command blocks are prepared in the controller's memory, and the first command block is passed to the controller hardware to be executed. Upon completion, an interrupt routine checks for errors, then directs the hardware to execute the next command block. When the last command block is finished, the callback function is invoked, passing a kind of summary error status to the user. If a 1553 error occurs during any command block execution, the rest of the command blocks are aborted, and the status is returned that includes the transfer index number that was active at the time the error occurred, along with the controller's error status word. The driver then returns to await the next message queue entry.

Driver parameters, status responses

Inside the driver is the main entry point called `drv1553Execute`, which is invoked by the EPICS driver support layer to effect 1553 data transfers. It captures the arguments passed to it and writes them into the message queue selected by the controller number. The arguments are as follows:

controller number, range 0–5
ptr to array of transfer specs
number of transfer specs
ptr to callback function that the driver invokes upon transfer completion
long word argument to be passed to callback function

Each transfer spec array element consists of the following structure:

1553 command word (16 bits)
number of bytes to transfer (16 bits)
ptr to data buffer to send or receive (32 bits)

Note that the number of bytes specified in a single transfer may exceed 64 bytes, the 1553 hardware limit of 32 words, in which case multiple command blocks will be built to support the complete transfer, each using the same command word.

The reason to permit multiple transfers in one call to `drv1553Execute` is that it may sometimes be necessary to insure that several commands are executed in sequence, without interruption, in order to accomplish some particular task. All transfers in one array specified in the call to `drv1553Execute` will be executed "atomically," and only a single invocation of the callback routine will result. Note that the user should not alter the contents of data buffers passed via `drv1553Execute` until the callback function is invoked. Only at that time can it be assured that output buffers are no longer needed and input buffers contain the data acquired from the hardware.

At this time the only returned error status from the call to `drv1553Execute` is –1, where the value available via `errno` is 1, which means Invalid controller#. In this case, nothing will be wwritten into the mesage queue, and the callback function will not be invoked. Additional software checks are made later during driver processing of the requested transfers. Error codes resulting from those checks are returned via the callback function.

The user callback function has two arguments and returns void.

long word argument as passed via `drv1553Execute` call
long word status

The long word of status may indicate several possibilities:

`00000000`	No errors. Transfer was successful.
`xfrIndex<<16 + errWord`	Transfer index# of err, controller error word.
`–1<<16 + err`	Software error detected, where `err` specifies error.

The values for `xfrIndex` range from 0 to the number of transfers minus one. The `errWord` value is furnished by the controller hardware. A likely cause may be a lack of timely response from an addressed RT. (The 1553 specification requires an RT to commence responding within 14 microseconds from reception of a command.)

The values for `err` that can occur are the following:

1	Invalid controller#
2	Bad argument value
3	Odd #bytes specified. Even #bytes required with 1553.
4	Invalid #bytes. Mode code without data must specify 0 bytes.
5	Too many command blocks. Overflowed space for command blocks
6	n.u.
7	Timeout awaiting interrupt from controller
8	Error from message queue access via mq_receive
9	Invalid message queue entry type#

The error specification for the `errWord` value are part of the 1553 standard:

Bit	*Meaning*
15–11	RT address (not used for controller)
10	n.u.
9	Improper sync
8	Address mismatch error
7	Improper word count

6	Response time error
5	Information field > 16 bits
4	n.u.
3	Invalid Manchester II
2	Parity error
1	n.u.
0	n.u.

Test vehicle for `drv1553Execute`

To assist in exercising the driver software, the routine `drv1553Command` is provided. It invokes the `drv1553Execute` entry point specifying a single transfer. Here are the arguments for `drv1553Command`:

controller number
RT address field
Subaddress field
First data word value
Number of data words

The RT address may specify any value from 0–31, where 31 is interpreted by the controller for broadcasting to all RT's. (This feature has not yet been used.) For Subaddress values of 1–30, if the transfer command is output (sending data to the RT), then the number of data words is specified > 0, and the data buffer is filled with an increasing sequence of 16-bit values starting with the first data word value. If the transfer command is input (reading words from the RT), then the number of words is specified as < 0, and the data words will be displayed upon completion of the transfer.

For Subaddress values of 0 or 31, used to imply a "mode code" command, no more than a single data word is used. In this case, the number of words parameter is used to specify the mode code value, in the range 0–31. Use of the various mode code values are standardized. Some of them, with mode code values of 0–8, do not use a data word at all. Those that do, with mode code values of 16–21, imply either an input data word or an output data word, but not both. Here is a brief list of the standardized mode code values:

Mode code	*T/R bit*	*Function*	*Data?*	*Broadcast?*
0	1	Dynamic bus control	N	N
1	1	Synchronize	N	Y
2	1	Transmit status word	N	N
3	1	Initiate self test	N	Y
4	1	Transmitter shutdown	N	Y
5	1	Override transmitter shutdown	N	Y
6	1	Inhibit terminal flag bit	N	Y
7	1	Override inhibit terminal flag bit	N	Y
8	1	Reset remote terminal	N	Y
9–15	1	(reserved)	N	TBD

16	1	Transmit vector word	Y	N
17	0	Synchronize	Y	Y
18	1	Transmit last command	Y	N
19	1	Transmit BIT word	Y	N
20	0	Selected transmitter shutdown	Y	Y
21	0	Override selected transmitter shutdown	Y	Y
22–31	1/0	(reserved)	Y	TBD

D0 Rack Monitor

One example of an RT is that used throughout much of the D0 detector for slow controls data acquisition. It interfaces to 8 D/A channels, 4 words of digital I/O and 64 channels of A/D. Here is a brief summary of the subadress values used:

Subaddress	*T/R*	*Description*
1	0/1	D/A chan 0
2	0/1	D/A chan 1
3	0/1	D/A chan 2
4	0/1	D/A chan 3
5	0/1	D/A chan 4
6	0/1	D/A chan 5
7	0/1	D/A chan 6
8	0/1	D/A chan 7
9–15	x	n.u.
16	0/1	Digital I/O (P3)
17	0/1	Digital I/O (P8)
18	0/1	Digital I/O (P2)
19	0/1	Digital I/O (P7)
20	1	A/D channels 0–63 (P5,P6,P10,P11)
21–29	x	n.u.
30	1	Status/ID word
31	0	Synchronize mode code (17): initial A/D chan#

Each D/A channel may be read or written. Each digital I/O word may be read or written, as determined via switches on the front panel. The RT address is also set via such front panel switches. The labels Px identify I/O connectors on the rear panel.

Additional driver entry points

As part of the driver package, the VxWorks-required routines `drv1553Init` and `drv1553Report` are available. As shown before, `drv1553Init` must be invoked before any drivers are initialized via `drv1553Define`. The `drv1553Report` routine merely lists out the current set of initialized drivers, one for each controller.

Each 1553 controller driver is initialized via `drv1553Define`, with these arguments:

controller number
interrupt vector number (8-bit)
message queue size—maximum number of entries available

VMEbus controller memory base address
Offset to start of command block area
Size of command block area

Complementary to `drv1553Define` is the routine `drv1553Undefine`, which is used to remove a 1553 controller driver. It performs this task by writing a special type of message into the message queue that is interpreted by the driver code as a signal to terminate itself and release all its reserved resources. The single argument for `drv1553Undefine` is the controller number.

Evolving system

Of course, this test environment herein described is one that continues to undergo change. Nonetheless, this description should serve to acquaint a new user who is interested in exercising the VxWorks 1553 driver software package. See Robert Goodwin or Fritz Bartlett for additional information.